

REMARKS

Applicant expresses appreciation to the Examiner for consideration of the subject patent application. This amendment is in response to the Office Action mailed November 1, 2005. Claims 1-21 were rejected. Claims 1-21 were originally presented. Claims 1-16 and 18-21 remain in the application. Claim 17 has been canceled. Claims 1-4, 7-11, 15-16, and 18-21 have been amended. No claims have been added.

Claim Objections

Claims 1, 9, and 18 were objected to for informalities. The Office Action states that there is a missing “the” before “remaining object code files”. Applicant has amended the claims accordingly.

Claim Rejections - 35 U.S.C. § 101

Claim 17 has been canceled and Application requests that the rejection under 35 U.S.C. § 101 be withdrawn.

Claim Rejections - 35 U.S.C. § 112

Claims 1-21 were rejected by the Office Action as being indefinite in light of the specification. In particular, the Office Action stated that “the limitation recited as ‘when the computer program is executed’ does not appear to associate a program being executed as commonly accepted and understood by one skill in the art in regard to compiler arts.”

In order to understand the present invention, a detailed explanation of what is described in the description will now be provided. The specification describes a compile-time preparation stage and then an actual execution stage. The present claims cover actual execution stage.

The compile-time stage is the definition of an identifier (e.g., a timestamp) and a storage location in a defined static storage location for a software class. The specification describes the example that a macro is first defined (for example, PIKA_TIMESTAMP and PIKA_HOWBUILT). See specification page 4, line 16 - page 5, line 11. A static class is created with storage locations for the identifier or timestamp. Specification page 6, lines 7-30. The

preprocessor is used to “hardcode” the identifier or timestamp into the source code that creates the static class and storage location. Specification page 5, lines 13-26.

A preprocessor is basically a textual search and replace before the actual compilation step takes place using a source code language compiler. In other words, the preprocessor sets the identifier or timestamp as a textual input for the static storage location in the CheckVersion class. A helpful resource for understanding preprocessing and static classes is the book “Thinking in C++ 2nd Edition” by Bruce Eckel which can be accessed for free at <http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html>. This preprocessing step is not covered in the independent claims, nor are the claims to be construed to cover any compile time processing.

The run-time portion of the invention uses a constructor that does the actual checking when the compiled application is executed at run-time. A constructor for the class is compiled into object code, along with the static storage location and timestamp in the class. However, the object code does not actually perform any comparison function until the final compiled program is executing on a processor. Specification page 7, line 31 – page 8, line 6. A constructor is generally called when a class is created. In the present invention, the constructor performs these steps at run-time:

“comparing the version of the selected object code file with a version of each of the remaining object code files of the plurality of object code files **when the computer program is executed**; and

generating an alert in response to the version of the selected object code file being different than one or more versions of the remaining object code files.”

The comparing and generating steps cannot take place at compile time because the constructor is not actively executing at that time.

The claims refer to the comparison of a version in the object code file which is not taught or suggested by the prior art. The prior art does **not teach or suggest an identifier or timestamp in an object code file**, as in amended claim 1. Checking object code files against each other protects against problems that are generated in multiple compiler builds at different

sites or by multiple software developers.

The present invention is not claiming a method that happens at compile time nor one that is performed by a compiler. The present invention occurs in object code files in a final executable that have been built by a compiler (**not in** an executing compiler process). For example, the claimed comparison may happen when a constructor is first called or at some other defined execution point in the application.

Claim Rejections - 35 U.S.C. § 103 – New Rejection - Part (D) - page 12

Applicant has described the interpretation of the term “executing program” with respect to 35 U.S.C. § 112 and believes the discussion above and the claim amendments overcome the overall indefiniteness rejection. In addition, claim 17 has been canceled in order to overcome the rejections with respect to claim 17. However, a description of why these arguments are not applicable to any other claims of the invention is also discussed below.

The Office Action has stated the concern that the claims and specification are unclear regarding whether “the claim amounts to a process leading or building up to but no necessarily execution per se of an executable.” Applicant has shown the specification is clear that the comparison step of the claims is made in the final executable and not at compile time or by the compiler executable. The claims have also been amended to include the term “object code file” to further make this distinction.

The Office Action asserted that Bowman identifies when two different versioned object code files have different versions. Bowman does not teach this. What Bowman teaches is using timestamps for logging and processing helpdesk inquiries (page 68, paragraphs 2165-2172). This is unrelated to run-time SDK version error detection in a final executable.

Ching is merely a web enabled software system for comparing one or more word processing documents and then showing the text changes within the document in side-by-side windows or panes. See Summary of Ching. Comparing two text documents in a web application does not teach or suggest “comparing the version of the selected object code file with a version of each of the remaining object code files of the plurality of object code files when the computer program is executed.”

Claim Rejections - 35 U.S.C. § 103 – Previous Rejections

Claims 1-16 and 18-21 (including independent claims 1, 9, and 18) were rejected under 35 U.S.C. § 103 as being unpatentable over Ching (USPN 5,560,620), hereafter “Ching” in view of Leblang et al. (USPN 5,649,200), hereafter “Leblang”.

The present invention, as claimed in amended independent claim 1, is a method for verifying a version of each of a plurality of object code files in a computer program at runtime. An object code file is machine code generated by a compiler after it processes a source code file. When different versions of software developer kits are used to generate object code files, subtle program errors can result when a computer program uses object code files having differing versions. Thus, the present invention enables a version number of the software developer kit to be placed within each object code file and enables each object code file in a computer program to be checked when the computer program is executed. The files are checked to determine that the object code files were generated by the same software developer kit as the other object code files that comprise the computer program. If comparisons show two or more versions, an error message can be generated alerting to the problem or execution of the program may be stopped.

Even if the structures of the Ching and Leblang references are combined as proposed by the Examiner, the result would necessarily constitute a structure different from that of the applicant and one that would not accomplish the result of the claimed invention. The Examiner states that the common endeavor of Ching and Leblang is “to have persisted versions of files that can be reused for subsequent application so that systematic rebuild of files without comparison would not be obviated, thus making the reuse effort less extensive.”

However, the result of the present invention, as claimed in claim 1, is not to have persistent versions of files that can be reused. Rather, as explained in the application, one embodiment of the present invention is to reduce or eliminate bugs in programs caused when different versions of software developer kits are used to compile programs. The present invention does not check to see if a version of an object code file has been previously compiled, as taught in Leblang. The previous compilation of an object code file is irrelevant to the present invention. Rather, claim 1 reads, in part:

comparing the version of the selected object code file with a version of each of remaining object code files of the plurality of object code files when the computer program is executed; and

generating an alert in response to the version of the selected object code file **being different** than one or more versions of the remaining object code files.

Thus, the object of one embodiment of the present invention is to assure that at computer program execution time the object code files in a computer program were all compiled using the same version of a software developer kit or compiler. Neither Ching nor Leblang, nor a combination of the two references teach or suggest this result. Using the teachings of Ching and Leblang, one skilled in the art would not be enabled to determine whether an executable computer program was comprised of object code files compiled by differing versions of a software developer kit. Therefore, Applicant respectfully submits that independent claim 1 is allowable, and urges the Examiner to withdraw the rejection.

The same arguments can be applied to independent claims 9 and 18, which are similar in structure to claim 1.

Rejection of the dependent claims 2-8, 10-16, and 19-21 should be reconsidered and withdrawn for at least the reasons given above with respect to the independent claims. The dependent claims, being narrower in scope, are allowable for at least the reasons for which the independent claims are allowable.

The combination of structures of the Ching and Bowman references as proposed by the Examiner would result in a structure that is necessarily different from the embodiments of the invention claimed by the Applicant, and one that would not accomplish the result of the claimed invention. Neither Ching nor Bowman, nor their combination teach or suggest a software development kit configured to facilitate a development of an application configured to generate an error when the application is executed in response to two or more of the object code files having different time stamps. Using the inventions taught or suggested in Ching and Bowman, it would not be possible to determine in an executable file whether the program executable

included object code files that were compiled using different software developer kits. Thus, it would not have been obvious for the Applicant to arrive at the present invention using the teachings or combination of Ching or Bowman.

CONCLUSION

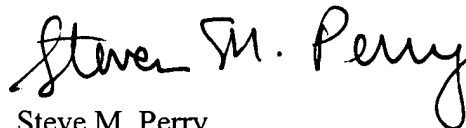
In light of the above, Applicant respectfully submits that pending claims 1-16 and 18-21 are now in condition for allowance. Therefore, Applicant requests that the rejections and objections be withdrawn, and that the claims be allowed and passed to issue. If any impediment to the allowance of these claims remains after entry of this Amendment, the Examiner is strongly encouraged to call Steve Perry at (801) 566-6633 so that such matters may be resolved as expeditiously as possible.

No claims were added. Therefore, no additional fee is due.

The Commissioner is hereby authorized to charge any additional fee or to credit any overpayment in connection with this Amendment to Deposit Account No. 08-2025.

DATED this 30th day of January, 2006.

Respectfully submitted,

A handwritten signature in black ink that reads "Steve M. Perry". The signature is written in a cursive, flowing style.

Steve M. Perry
Registration No. 45,357

THORPE NORTH & WESTERN, LLP
Customer No. 20,551
P.O. Box 1219
Sandy, Utah 84091-1219
Telephone: (801) 566-6633